Fast Lookup Tables for Interatomic Interactions

NATHAN G. HUNT[1*] and FRED E. COHEN[1–4]

[1]*Department of Pharmaceutical Chemistry,* [2]*Pharmacology,* [3]*Biochemistry and Biophysics, and* [4]*Medicine, University of California, San Francisco, California 94143-0446*

Received 12 January 1996; accepted 4 April 1996

ABSTRACT

We describe novel lookup tables for the rapid calculation of interatomic interactions. The tables have nonuniform distributions of bin widths tailored to minimize numerical error and maximize computational speed. Since interaction energies are precalculated, computer time requirements are essentially independent of the form of the potential function used. In test calculations using the AMBER force field and an internal coordinate Monte Carlo algorithm, the lookup table runs 15% faster than direct calculation of nonbonded interactions. The method is more advantageous for more complicated energy functions. As an example of a more complicated potential function, we have tested a pairwise approximation to accessible surface area. In this case, the use of the lookup table results in a speedup of a factor of two. The method is straightforward to implement and should be widely applicable. © 1996 by John Wiley & Sons, Inc.

Introduction

The calculation of nonbonded interactions is generally the most computationally intensive part of any molecular mechanics simulation. Since many applications of molecular simulations are limited by computer time requirements, several approaches have been devised to increase the speed of evaluating these interactions.

*Author to whom all correspondence should be addressed. E-mail: hunt@babar.ucsf.edu

One approach is to tailor the functional form of the interactions, choosing a function which can be evaluated relatively quickly. For example, this is one of the principal motivations for representing steric repulsion as proportional to r^{-12}, since it can be rapidly calculated from r^{-6}. The use of distance-dependent dielectrics, $\varepsilon = kr$, also provides this type of advantage, since it obviates the need to calculate square roots.

Another approach which works well for the calculation of protein–ligand interactions is the use of a three-dimensional grid of values of potential fields.[1,2] The use of this scheme is limited by the time and memory requirements involved in setting-up the grid. This method works very well

CCC 0192-8651 / 96 / 161857-06

for protein–ligand interactions in which the protein is regarded as rigid, providing stable potential fields in a limited region of space.

Two further approaches which show promise but have not yet been widely implemented in computational chemistry are the fast multipole method[3] and the use of a Fourier–Green's function.[4,5]

In this work, we present a simple lookup table scheme for calculating interatomic interactions. This method has some conceptual points in common with both the grid-based method and the fast multipole method. A lookup table is, in essence, a one-dimensional grid, indexed by the interatomic distance or, in our case, a simple function of the distance. As is the case with the fast multipole method, our lookup table makes use of the fact that the shorter distance interactions are stronger than longer distance ones, and therefore must be calculated with greater accuracy. Unlike the fast multipole method, which is $O(N)$ even without a cutoff scheme, the present method is $O(N)$ if a cutoff scheme is used, and $O(N^2)$ without one. Its principal advantage over the fast multipole method is greater ease of implementation.

Computational Methods

A conceptually straightforward implementation of a lookup table would use an index proportional to the interatomic distance. Thus, the calculation of an interatomic interaction would involve first calculating the interatomic distance, converting the distance to an integer index, and then recalling a value from a precalculated table. In C code, one could write:

```
index = (int) (k*sqrt(rsq));
energy = table[atom_type1][atom_type2][index];
```
(1)

where `(int)` is the C command for "take the greatest integer in," `k` is a constant which sets the grid spacing or bin size of the table, `sqrt` is the square root function, `rsq` is the square of the interatomic distance, `table` is a precalculated three-dimensional array of interaction energies, and `atom_type1` and `atom_type2` are integers corresponding to the two types of atoms being considered.

The implementation given above suffers from two drawbacks. One is the need to calculate a square root, which is relatively time-consuming. The other is the fact that the table might have to be quite large to accommodate long-distance interactions and to simultaneously attain a small enough bin size that accuracy is not degraded.

The need to calculate a square root can be eliminated by trying:

```
index = (int) (k*rsq);
energy = table[atom_type1][atom_type2][index];
```
(2)

The values stored in this table correspond to energies evaluated at unevenly spaced distances. The bin size is largest for separations near zero and becomes smaller with distance. Thus, to have sufficiently small bin sizes for short distances, the constant `k` and the size of the table have to be made relatively large. Compared with the table given by the expressions in (1), the expressions in (2) have eliminated the square root at the expense of exacerbating problems with numerical accuracy and memory requirements.

We can retain some portion of the speed increase obtained in expressions (2) and reduce the memory and accuracy limitations by calculating the key to the lookup table so that the bin widths are relatively small in those distances ranges where the interatomic energies are changing the most rapidly. This idea is made more precise below. We have explored three ways of calculating such an index, given by the C expressions (3)–(5):

```
index = (int) (A / (B + rsq));          (3)
index = (int) ((C - D*rsq) * rsq);      (4)
```

```
union {
    float flt;
    int intgr;
   } rsq;

rsq.flt += 2.0f;
rsq.intgr >>= 14;
index = rsq.intgr & 4095;               (5)
```

where `A`, `B`, `C`, and `D` are constants.

With expression (3), the value of the index decreases monotonically with increasing distance. The bin size is smallest for distance $(B/3)^{1/2}$. By choosing `B` appropriately, the bin size can be made small for those distances which are most important

for obtaining accurate results (see below). Figure 1 plots the bin width as a function of distance for a table with 4096 entries and a minimum bin size at 1.7 Å of separation.

Expression (4) is monotonic only for small values of `rsq`. Therefore, it can only be used for systems of limited size or with a distance-based cutoff scheme. The bin size has a broad minimum, with its smallest value at 8.1 Å. The bin width as a function of distance is shown in Figure 2.

Expression (5) uses the `union` construct, in which the same memory location can be utilized to store different data types. The first four lines in expression (5) are the declaration of the `union` and the last three lines are the executable code. In FORTRAN, an `EQUIVALENCE` statement could be used in place of the `union`. Here we use the `union` to store the floating-point representation of the square of the interatomic distance. The index for the lookup table is then obtained by manipulating the bits of the floating point number to obtain an integer within the required range. >> = is the C representation of the right-shift bit operation, and & is the C representation for the logical "and" of the bits of the two operands. The code given in expression (5) assumes the ANSI/IEEE Standard 754—1985 representation for a 32-bit floating point number.[6,7] Expression (5) converts floating point numbers in the range 0–22.5 to integers between 0 and 4096, the table size we most often use. To obtain a table of size 8192, the 14 is changed to a 13 and the 4095 to 8191.

Expression (5) essentially converts a floating point number into a much lower precision floating point number. A single precision floating point number has a sign bit, eight exponent bits, and 23 fractional bits. Expression (5) can be thought of as reexpressing a floating point number in a representation with no sign bit, three exponent bits, and nine fractional bits.

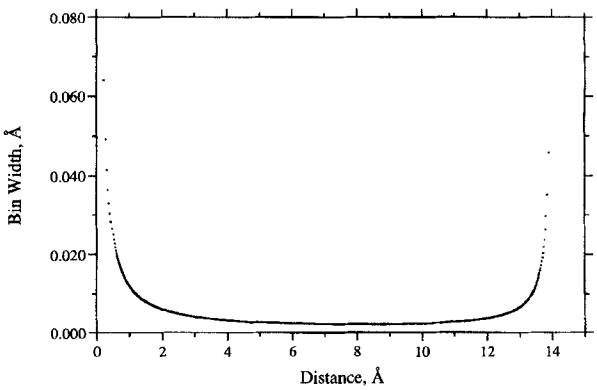

FIGURE 2. Bin width as a function of interatomic distance distance for the indexing scheme of expressions (4). The table size is 4096; C = 41.8, D = 0.107.

The addition of 2.0 to the distance in expression (5) is necessary to obtain a monotonic function from the floating point representation. Floating point numbers smaller in magnitude than 2.0 have negative exponents in the standard representation, and so would not fit smoothly into the scheme given in expression (5). The addition of 2.0 also results in a reasonably good distribution of bin widths, as shown in Figure 3. Using `2.0f` instead of `2.0` avoids time-consuming type conversions from single to double precision and back. As in the case with expression (4), the finite range of values of `rsq` which work with expression (5) restricts the

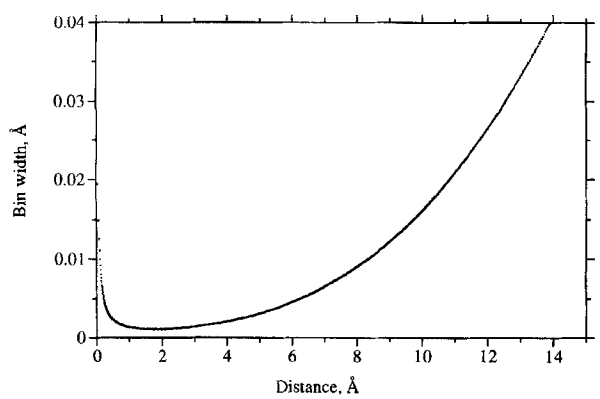

FIGURE 1. Bin width as a function of interatomic distance for the indexing scheme of expressions (3). The table size is 4096; A = 36,864, B = 9.0.

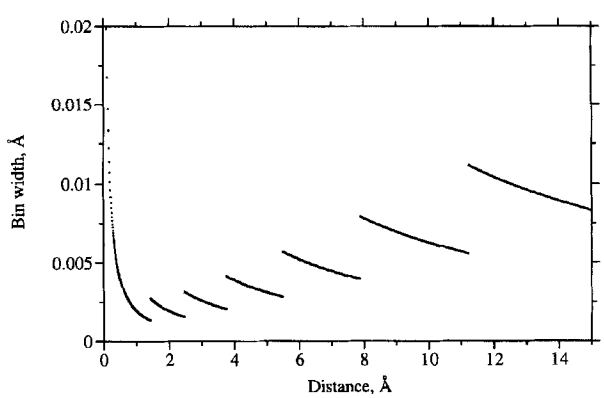

FIGURE 3. Bin width as a function of interatomic distance for the indexing scheme of expressions (5). The table size is 4096.

use of this scheme to small systems or to calculations which use a distance cutoff.

We have included a discussion of expressions (3), (4), and (5) here to demonstrate that a variety of approaches may be used to tailor bin size as a function of interatomic distance and to simultaneously achieve computational speed. We have used expression (5) for our calculations, as discussed below.

AN IMPLEMENTATION DETAIL

In all of the aforementioned approaches, the table of nonbonded interaction energies is a large array containing $N_T(N_{\text{atom types}})^2$ floating point numbers, where N_T is the number of distance bins and $N_{\text{atom types}}$ is the number of atom types being used. For protein simulations, we have generally used 4096 for N_T and 29 for $N_{\text{atom types}}$, giving an array of 3.4×10^6 floating point numbers, requiring 11 megabytes of memory. While an array of this size fits comfortable within the main memory of current workstations, it does not generally fit into the data caches, which are smaller and faster than main memory. Therefore, accessing entries of the lookup table is likely to result in cache misses, slowing down program execution.[8] Delays can be avoided by writing the code so that unrelated instructions are executed while lookup table values are being fetched from main memory.

NUMERICAL ERROR

We can write an expression for the error in the energy evaluation, ΔE, due to the use of a lookup table:

$$|\Delta E| \leq \Sigma_{\text{bins}} n_{\text{bin}} \, dE/dr|_{\text{max,bin}} \, \delta r_{\text{bin}} \qquad (6)$$

where the sum is over the distance bins [or, equivalently, over the values of `index` in expressions (3), (4), or (5)], n_{bin} is the number of interatomic distances falling within the bin, r is the interatomic distance, $dE/dr|_{\text{max,bin}}$ is the maximum absolute value of the derivative of the energy within a given bin, and δr_{bin} is the width of a given bin.

Two important properties of the terms in eq. (6) are that the energy derivative tends to zero for large values of r, and that, unless the conformation of the molecule is far from relaxed, n_{bin} tends to zero for small values of r, due to steric interactions. Thus, the largest part of ΔE should be contributed by bins within some intermediate distance range. By choosing the distribution of bins such that the smallest bin sizes are within this range, $|\Delta E|$ can be minimized. All three schemes discussed here, expressions (3)–(5), have minimum values of bin widths for some intermediate range of distance, as shown in Figures 1–3.

If a constant dielectric is used and the system under consideration has groups of atoms which carry net charges, the reasoning presented above may not hold for long-range interactions between charged groups. That is, the energy derivative in eq. (6) may not go to zero sufficiently rapidly to permit the use of larger distance bins for long-range interactions. In this case, the lookup tables could be supplemented by a separate evaluation scheme for charge–charge interactions.

The lookup table schemes presented here can be considered in the context of information and coding theory.[9] In the terminology of these fields, our lookup tables are codes for the interaction energies, and the numerical error given by eq. (6) is a measure of the distortion of the code. The rate of the code is the log base 2 of the size of the table. Thus, what we have done is to find codes which offer low distortion at a relatively low rate, and which can be decoded efficiently.

Results and Discussion

We found that expression (5) gave the best performance of the three lookup tables presented here. Expression (3) ran slightly slower than expression (5), and expression (4), which required a larger lookup table to obtain comparable accuracy, was the slowest of the three. The relatively large bin sizes in the distance range around 2 present a problem for the use of expression (4). This could be ameliorated by using an offset:

```
rsq = rsq + OFFSET;
index = (int) ((C - D* rsq) * rsq);
```
(7)

A value of OFFSET of around 2.0 should result in a sufficiently dense spacing of bins for small interatomic distances.

Table I summarizes the results of test calculations of an alanine 40-mer using an internal coordinate Monte Carlo algorithm and three different energy functions, comparing the performance of the lookup table expression (5) with straightforward calculation of interatomic interactions. Calculations with valine and threonine 40-mers give

TABLE 1.
Comparison of running Time and Numerical Error in Energy Evaluation for Lookup Table Index Described by Expressions (5) and for Direct Calculation.[a]

Method of Calculation	Energy function	Average time required	Absolute error in energy (kcal / mol)
Lookup table	Eq. (8)	3450	0.21
Lookup table	Eq. (9)	3397	0.25
Lookup table	Eq. (10)	3492	0.22
Direct calculation	Eq. (8)	3217	0.15
Direct calculation	Eq. (9)	3989	0.06
Direct calculation	Eq. (10)	6462	0.07

[a]An internal coordinate Monte Carlo algorithm was run for 500,000 energy evaluations of a polyalanine 40-mer. Three different energy functions were used. All values are averages over the results of 10 runs. The lookup table size was 4096.

very similar results. The energy functions used were:

$$E = \Sigma_{i,j} A_{ij} r_{ij}^{-12} - B_{ij} r_{ij}^{-6} + 332 q_i q_j / 4 r_{ij}^2 \quad (8)$$

$$E = \Sigma_{i,j} A_{ij} r_{ij}^{-12} - B_{ij} r_{ij}^{-6} + 332 q_i q_j / 4 r_{ij} \quad (9)$$

$$E = \Sigma_{i,j} A_{ij} r_{ij}^{-12} - B_{ij} r_{ij}^{-6} + 332 q_i q_j / 4 r_{ij} + C_{ij}(r) h_i + C_{ji}(r) h_j \quad (10)$$

$$C_{ij}(r) = \pi (a_i + a_w)(a_i + a_j + 2a_w - r_{ij}) \times \left(1 + (a_j - a_i)/r_{ij}\right) \text{ if } r_{ij} < a_i + a_j + 2a_w$$
$$= 0 \text{ otherwise}$$

where A_{ij}, B_{ij}, and q_i are the van der Waals parameters and atomic charges from the AMBER force field,[10] r_{ij} is the interatomic distance, a_i are the AMBER van der Waals radii, a_w is a solvent probe radius set to 1.4 Å, and h_i are atomic solvation parameters.[11] $C_{ij}(r)$ is a pairwise approximation to accessible surface area.[12]

The lookup table is somewhat slower than straightforward calculation when eq. (8) is used and no square roots need to be calculated. Our code ran 7% slower with the lookup table than without it. Profiles of the time spent on different portions of the code indicate that about 61% of the running time was spent on calculating interatomic interaction. Thus, we infer that the lookup table scheme is 11% slower than straightforward calculation of the interactions is this case.

If eq. (9) is used, direct calculation of the energy requires square roots and the lookup table scheme is faster. The running time given in Table I is 15% faster with the lookup table than without it. Based on profiles of the code, we infer that the lookup table yields a 22% speed-up in calculating the interactions in this case.

The lookup table is of greatest advantage for complicated energy functions such as eq. (10). In this case the code runs 2.0 times faster than direct calculation, and the evaluation of nonbonded interactions is 2.4 times faster.

The effects of varying the lookup table size are illustrated in Table II. Smaller tables give faster execution times at the cost of increased numerical error. On our workstations, speed of execution falls off rapidly for tables of size greater than 8192.

Several types of criteria can be used to evaluate whether the numerical errors arising from the use of the lookup table are tolerable. An error of 0.25 kcal/mol may seem relatively large compared to free energies of experimental interest. For accurate calculations of free energies, larger table sizes may be necessary, although we note that the relative error in the difference in energies between conformations may be smaller than the absolute

TABLE 2.
Running Time and Numerical Error in Energy Evaluation As a Function of Table Size.[a]

Size of lookup table	Average time required	Absolute error in energy (kcal / mol)
2048	3224	0.53
4096	3397	0.25
8192	3805	0.09

[a]The indexing scheme of expressions (5) was used with the energy function given by eq. (9). All values are averages of ten runs of 500,000 energy evaluations of a polyalanine 40-mer.

error reported here. Another way of evaluating the error is to consider the performance of the algorithm. For example, one could check whether the use of the lookup table results in essentially the same ensemble of states as direct evaluation of the interactions. We have used a related criterion—whether the algorithm using the lookup table generates low-energy conformations. Based on this criterion, we have found its performance entirely satisfactory.

We have investigated the use of lookup tables with Monte Carlo algorithms. While the lookup tables could possibly be applied to molecular dynamics, we have not investigated this application. Molecular dynamics is considerably less tolerant of numerical error than is Monte Carlo. The lookup tables described here could be combined with interpolation schemes for use in molecular dynamics, but the additional calculations involved would reduce any gain in speed.

Conclusions

The lookup table methods presented here provide two related advantages over direct calculation of interaction energies. One is an increase in computational speed for the most commonly used functional form for nonbonded interactions, eq. (9). The other is the fact that computational time requirements are essentially independent of the form of the interaction used. This frees the user of the need to tailor the form of the interaction for the sake of computational speed. These advantages should make the methods widely applicable.

Acknowledgments

The authors thank Steve Sizemore for computer support. This work was supported by a grant from the National Institutes of Health (GM39900).

References

1. B. Busetta, I. J. Tickle, and T. L. Blundell, *J. Appl. Cryst.*, **16**, 432 (1983).
2. E. C. Meng, B. K. Shoichet, and I. D. Kuntz, *J. Comput. Chem.*, **13**, 505 (1992).
3. L. Greengard and V. Rokhlin, *J. Comput. Phys.*, **73**, 325 (1987).
4. A. M. Mathiowetz, A. Jain, N. Karasawa, and W. A. Goddard III, *Proteins*, **20**, 227 (1994).
5. R. W. Harrison, I. V. Kourinov, and L. C. Andrews, *Prot. Eng.* **7**, 359 (1994).
6. G. Kane and J. Heinrich, *MIPS RISC Architecture*, Prentice-Hall PTR, Upper Saddle River, NJ, 1992.
7. R. P. Paul *Sparc Architecture, Assembly Language Programming*, & C, Prentice-Hall, Englewood Cliffs, NJ, 1994.
8. K. Dowd, *High Performance Computing*, O'Reilly & Associates, Sebastopol, CA, 1993.
9. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, 1991.
10. S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, and P. Weiner, *J. Am. Chem. Soc.*, **106**, 765 (1984).
11. D. Eisenberg, M. Wesson, and M. Yamashita, *Chemica Scripta*, **29A**, 216 (1989).
12. S. J. Wodak and J. Janin, *Proc. Natl. Acad. Sci.*, **77**, 1736 (1980).